

# Exploration, Analysis, and Manipulation of Source Code using srcML

Jonathan I. Maletic  
Department of Computer Science  
Kent State University  
Kent, Ohio 44242  
jmaletic@kent.edu

Michael L. Collard  
Department of Computer Science  
The University of Akron  
Akron, Ohio 44325  
collard@uakron.edu

**Abstract**— This technology briefing is intended for those interested in constructing custom software analysis and manipulation tools to support research or commercial applications. srcML (srcML.org) is an infrastructure consisting of an XML representation for C/C++/C#/Java source code along with efficient parsing technology to convert source code to-and-from the srcML format. The briefing describes srcML, the toolkit, and the application of XPath and XSLT to query and modify source code. Additionally, a hands-on tutorial of how to use srcML and XML tools to construct custom analysis and manipulation tools will be conducted.

**Index Terms**—srcML, static program analysis, program transformation, XML.

## I. INTRODUCTION

**srcML** (sōrs em el), *n.* **1.** an infrastructure for the exploration, analysis, and manipulation of source code. **2.** an XML format for source code. **3.** a lightweight, highly scalable, robust, multi-language parsing tool to convert source code into srcML. **4.** a free software application licensed under GPL.

Conducting research in software engineering and maintenance very often requires the analysis and manipulation of large bodies of source code. Researchers need to apply static analysis techniques, compute program slices, calculate software metrics, or search for a specific pattern to solve more complex problems such as feature location or impact analysis. Software evolution also implies transitioning the current system to a new version in a (hopefully) systematic manner. Thus, the ability to apply refactorings and transformations to a subset or the entirety of a code base is necessary for undertaking more complex problems (e.g., large adaptive changes for a new API or library).

Unfortunately, the amount of engineering involved to build a program slicer or apply a transformation on a 5 MLOC system is often quite daunting for the freshman and well-seasoned researcher alike. The variety and types of programming languages used in large-scale software in industry further complicates matters. Languages such C/C++ are notoriously difficult to parse and make engineering even simple tools very arduous. Because the engineering is difficult, often only a partial or approximate solution is produced. The result is rarely a robust or flexible tool.

This was the main motivation behind our development of the *srcML* [1-4] infrastructure (www.srcML.org). Our goal was to construct a flexible and scalable tool that supports

various research needs in the exploration, analysis, and manipulation of source code. The infrastructure has recently undergone a number of substantial enhancements thanks to support from the US National Science Foundation.

The srcML infrastructure consists of two base elements, an XML representation and a toolkit for converting source code to-and-from the format. The srcML format is ideal for exploration and transformation tasks as it is an exact representation of the source code as written by the developer. All original source, including comments, preprocessor statements, and even white space, is perfectly preserved in the srcML representation. It should be stressed that parsing in the presence of preprocessor statements presents quite a challenge. From the srcML format, the original source code can be extracted, permitting round-trip transformations without loss of any formatting and coding style. The srcML format wraps the text of the source code in XML elements that mark the inherent syntax. The names of the elements reflect the programmer's view with tags such as `<if>`, `<function>`, `<class>`, etc.

The srcML toolkit supports the translation of C, C++, C#, and Java source to the srcML format. The self-contained parsing technology is robust and highly scalable both in time and space. The parser supports translation of single files, and even code fragments, e.g., a single statement. Entire source code projects can be stored (and then analyzed or transformed) in a single srcML archive. Single file translation to the srcML format is at 35KLOC/sec and supports multithreaded parsing, which scales that translation speed according to the number of cores. For example, the entire Linux kernel (over 35,000 source files) can be converted to the srcML format in less than 2 minutes on a 6-core desktop machine, reaching translation speeds of over 150 KLOC/sec. Conversion back to source code from the srcML format is even faster. The size of the resulting srcML file is ~4.5 times that of the original code and can be easily compressed, leading to a size of ~1.5 times that of the compressed original code.

The srcML infrastructure is freely available for download and licensed under GPL. Complete documentation on the format and language feature support is provided on srcML.org. We also provide executables for MS Windows, Mac OS, and various GNU Linux distributions.

## II. IMPORTANCE TO SOFTWARE ENGINEERING COMMUNITY

The objective of this technical briefing is to disseminate information about the recent enhancements to the srcML infrastructure to a broad cross section of the software engineering community. It is of particular interest to graduate students and researchers that need to construct custom exploration, static analysis, or transformation tools that are scalable to large code bases.

The srcML infrastructure can support research efforts on such things as architectural design recovery, programming language research, refactoring software to better utilize parallel hardware (e.g., GPUs), software reuse, large scale adaptive changes, and enterprise wide system analysis. It is also one of the few tools that support analysis across multiple languages.

srcML is already being used by a wide variety of researchers, including those in the fields of software engineering, programming languages, parallel and distributed processing, and computer science education. srcML has been used in the dissertation/thesis research of over two-dozen (and counting) computer science graduate students across a number of institutions. Additionally, it is being used by multiple commercial organizations.

In addition to the dissemination of information on the srcML infrastructure, another main purpose of this technical briefing is to obtain feedback from the research community on the srcML format and how the infrastructure can be improved to meet their needs.

## III. DESCRIPTION OF BRIEFING

This 90 minute briefing is organized into three main parts:

- An introduction and overview of srcML;
- How to build tools using the srcML C API and the srcSAX framework;
- A hands on tutorial of building a custom source code analysis tool.

The first part is a general introduction to srcML and the toolkit. Basic details of the implementation and scalability will be covered. The command line client will be presented and demonstrated. It allows users to easily apply XPath and XSLT to directly query source code. A number of examples will be presented. The goal is to demonstrate the ease of use and capabilities of the infrastructure along with providing an understanding of the types of problems srcML can tackle.

The second part of the briefing discusses how to integrate srcML into other tools via a C API called *libsrcml*. It allows direct calls to the parsing technology so that srcML can be seamlessly integrated into other visualization or analysis tools. Additionally, a C++ framework call *srcSAX* will be introduced that greatly reduces the learning curve to building custom analysis and transformation tools using srcML.

The last part of the technical briefing involves a hands-on tutorial of using the srcSAX framework to construct an actual analysis tool. This part of the briefing is supported with online

materials and code so that participants can follow along. There is also a web service for srcML that allows participants to parse source code without having to install software locally.

## IV. BIOGRAPHIES

*Jonathan I. Maletic* is Professor in the Department of Computer Science at Kent State University. He received the Ph.D. and M.S., both in Computer Science, from Wayne State University in 1995 and 1989 respectively. He received the B.S. in Computer Science in 1986 from The University of Michigan-Flint. His research interests are centered on software evolution, with a focus on the comprehension, analysis, manipulation, transformation, reverse engineering, traceability, and visualization of large-scale software systems. Prof. Maletic has authored over 100 refereed publications and is regularly funded by the US National Science Foundation. He has graduated twelve doctoral students, ten of which currently hold academic positions.

*Michael L. Collard* is an Assistant Professor in the Department of Computer Science at The University of Akron in Ohio, USA. He received a Ph.D. in Computer Science from Kent State University in 2004. He also received a M.S., and B.S. in Computer Science from Kent State University. His research interests focus on software evolution with 36 refereed publications, including a Most Influential Paper (MIP) Award, in the areas of source-code representation, analysis, transformation, and differencing. He is currently funded by the US National Science Foundation.

## ACKNOWLEDGMENT

This work was supported in part by a grant from the US National Science Foundation CNS 13-05292/05217.

## REFERENCES

- [1] Collard, M. L., Decker, M., and Maletic, J. I., "Lightweight Transformation and Fact Extraction with the srcML Toolkit", in Proceedings of 11th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM'11), Williamsburg, VA, USA, Sept 25-26 2011, pp. 10 pages.
- [2] Collard, M. L., Decker, M., and Maletic, J. I., "srcML: An Infrastructure for the Exploration, Analysis, and Manipulation of Source Code", in Proceedings of 29th IEEE International Conference on Software Maintenance (ICSM'13) Tool Demonstration Track, Eindhoven, The Netherlands, Sept. 22-28 2013, pp. 1-4.
- [3] Collard, M. L., Kagdi, H. H., and Maletic, J. I., "An XML-Based Lightweight C++ Fact Extractor", in Proceedings of 11th IEEE International Workshop on Program Comprehension (IWPC'03), Portland, OR, May 10-11 2003, pp. 134-143.
- [4] Collard, M. L., Maletic, J. I., and Robinson, B. P., "A Lightweight Transformational Approach to Support Large Scale Adaptive Changes", in Proceedings of 26th IEEE International Conference on Software Maintenance (ICSM'10), Timisoara, Romania, Sept 12-18 2010, pp. 10 pages.